Report No. 360

ILLIAC IV
QUARTERLY PROGRESS REPORT
April, May, and June 1969

Contract No.
US AF 30(602)4144

ILLIAC IV Doc. No. 230

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

ILLIAC IV
QUARTERLY PROGRESS REPORT
April, May, and June 1969

Contract No.
US AF 30(602)4144

Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois
61801

August 1, 1969

This work was supported in part by the Department of Computer Science, University of Illinois, Urbana, Illinois, and in part by the Advanced Research Projects Agency as administered by the Rome Air Development Center, under Contract No. US AF 30(602)4144.

TABLE OF CONTENTS

REPORT SUMMARY

Several major problem areas were discussed in the QPR for October-December 1968. As indicated in the QPR for January-March 1969, actions to solve these problems have been initiated. The contract with Fairchild to produce the semi-conductor memories has been signed and progress to date is according to schedule. A contract has been signed with Texas Instruments to provide the semi-conductor devices for the ILLIAC IV. Obtaining artwork for the production of the 12 layer CU boards remains a serious problem. Progress has been made in using the Design Automation programs to route the CU boards. However, Burroughs must increase the number of boards routed each week to insure that the CU schedule is met. The route program is being modified to reduce the computer time required to automatically route boards.

A PDP-9 computer has been ordered to attach to the PE exerciser to provide the ability to automatically diagnose errors in the PE and CU boards during manufacturing and final operation at the University. The PDP-9 will be delivered to the University in August for software development. In October the PDP-9 will be shipped to Burroughs to be interfaced with the PEX.

The monthly project review at Burroughs has proven to be very worthwhile, and the reviews will be continued. This quarter a new version of the machine characteristics and program manual has been drafted by Burroughs. This manual contains instruction timing information that was lacking in the original version. The published document will be available next quarter. In addition, the assembly language manual and simulator manual have become available. A user's group (FOURUM) was established for intercommunication and exchange of computer programs and documents for ILLIAC IV. A monthly newsletter will be sent to each participant.

As a result of the many changes in the project, Burroughs is now in the process of rescheduling their portion of the ILLIAC IV project. The new schedule will be available in August and some

slippage is anticipated. The University is in the process of renegotiating the Burroughs subcontract to reflect the changes in cost, schedule, and performance. Every effort will be made to identify milestones which will permit the University and Burroughs to begin negotiation on a fixed ceiling contract.

A project has been initiated to permit Burroughs ALGOL to be compiled and executed on CDC 6000 series machines. In particular, this provides 6000 series owners with the ability to simulate ILLIAC IV programs. Many potential ILLIAC IV users are currently users of the CDC 6000 series machines. To provide archival storage, an evaluation of a 10^{12} bit storage device was initiated.

A proposal to create a Center for Advanced Computation based on the ILLIAC IV computer has been presented to the Board of Trustees, University of Illinois. The main purpose of this Center is to provide an environment for the interdisciplinary use of the ILLIAC IV computer. Professor D. L. Slotnick is proposed Director of the Center. The Center staff will be primarily ILLIAC IV project personnel with additional staff being hired as the machine operation begins. It is anticipated that a new building will be constructed to house the personnel and the ILLIAC IV computer. Additional details will be presented in the next Quarterly Progress Report.

1.1 Logic Debugging and Diagnostics

1.1.1 PE Logic Simulation and Debugging

Verification of the corrected PE design and debugging of Burroughs' functional test routines were the main uses of the PE Logic Simulator, during this quarter. The power of parallelism was fully demonstrated. Because up to 47 cases are simulated in parallel, the execution of the simulator took only a fraction of time required by the PEXTAP assembler. The Equation Generator Program was used in updating equations of the control logic and generating inputs to the control logic test generator programs.

By the end of this quarter, Burroughs had updated the PE design with the University's recommendations and other corrections. A new simulator will be generated from the new wire list file in the next quarter.

1.1.2 CU Logic Simulator System

1.1.2.1 CU Card Simulator

Several improvements and corrections of the CU Card Simulator Generator system have been made using corrected net lists. A control program has been developed in order to execute various programs of the system automatically. It will be incorporated into the system soon with the history generation program.

1.1.2.2 TESLA

Debugging of the TESLA compiler has been completed. With the exception of the LOOP and INOUT verbs, the full syntax of the language has been implemented. Some minor modifications to the syntax have been made as a result of "living with" the language during the compiler writing and debugging phases. When an attempt was made to apply TESLA to PE card simulations, several differences between CU and PE signal naming rules

were discovered. Since signal names are recognized by their structure (e.g., 10 characters, presence of dashes, etc.), a few minor changes were necessary for TESLA to produce PE card simulation control files. The ILLIAC IV Translator Writing System has continued to be of invaluable assistance. It was felt that TWS assisted even during debugging. Error isolation was much easier because of the sharp separation between syntax and the various semantic routines.

1.1.2.3 Assembled Code Translator (ACT III)

The purpose of this program is to translate from the PEXTAP object code into a suitable input file for the CU Card Simulator and also to produce the printer control file for the simulator. In essence, ACT III is a simulator of the PEX and the CU Card Adapter. Part of this program is derived from the ACT program written for the PE simulator.

The program first loads the expected response, mask data, and microsequence registers of the PEX. It then decodes the EIC (Exerciser Instruction Command) into a time sequence of commands. At any given clock pulse, it compares the test patterns (microsequences) with the corresponding ones at the previous clock pulse and passes the changes, if any, along with the particular simulator input array subscript, input verb, storage element state, and various other commands to the simulator input file. The printer control file is formed using the expected response, the output array subscripts corresponding to the group of output pins of the simulator, and the various other printer commands.

The program uses the adapter card data and the simulator data to establish a correspondence between the PEX output signals and the simulator input subscripts. (Similarly, the correspondence between the output pins and the output array subscripts of the simulator is established.) Due to the lack of information of these data, the program had to be run only with simple examples. It appears that the PEXTAP files will have to be slightly modified to take into account the coordination of the input and output select bits required for the CU card tester. It is proposed to extend the ACT III program to provide output in a parallel form to the simulator.

1.1.3 PE Diagnostics

1.1.3.1 Path Tests

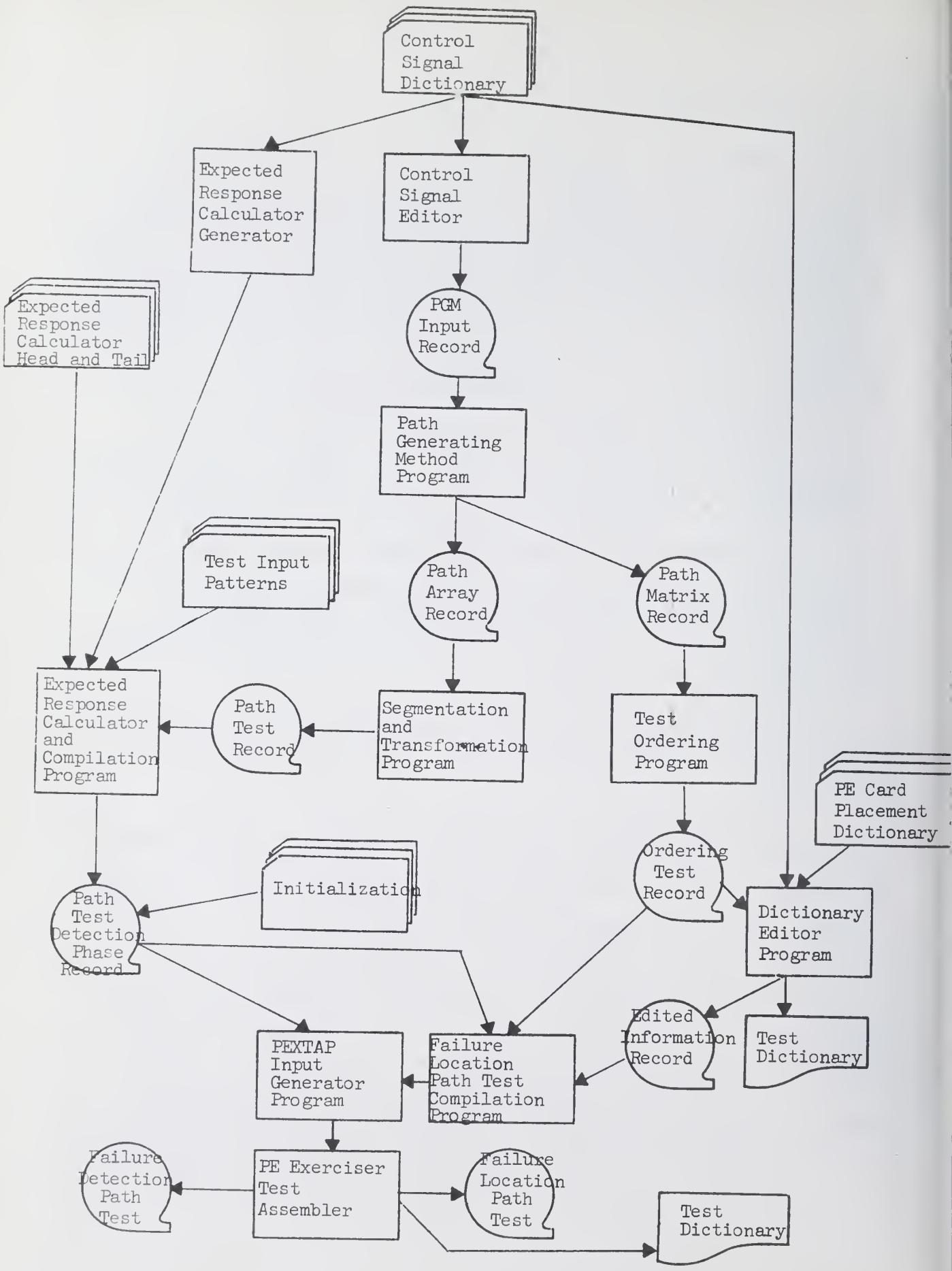
The path test generator system was completed in this period. Two major path tests have been generated: the failure detection path test and the failure location path test. The failure detection path test will be used for checking out the PE with the PE Exerciser to determine if there are component and/or wiring failures on the major data paths. If a failure is detected, then the failure location path test is executed to locate the failure within a small area of the machine.

An earlier version of the failure detection path test was released at the beginning of this period, and the first version of error location path test was created with the preliminary test dictionary.

The path test generator system requires two major inputs, the control signal dictionary and the PE card placement dictionary, both of which are written in fixed format. The control signal dictionary is translated into an arc description of the major data paths for the PGM (Path Generating Method) program.

The PGM finds a minimal set of test paths on the system described in a directed graph and creates a set of test paths and a path matrix.

The segmentation and transformation program reduces the dummy nodes which have been temporarily inserted in the graph model of the PE data flow and segments the test paths at appropriate points (for example, if a test path passes through a same node twice, it should be segmented at a storage element in the loop). The expected response calculator and compilation program (which is essentially a functional simulator) is generated from the control signal dictionary with additional routines. This produces the input file of the failure detection path test for the PEXTAP (PE Exerciser Test Assembly Program) except the microsequences which must be translated into the input format by the PEXTAP Input Generator Program. Each path test string follows a fixed initialization micro-step



PE Off-line Path Test Generator System

to clear registers. The expected responses are verified twice by the logic simulator for the output file of the expected response calculator and the input file to the PE exerciser. The failure detection path test provides no information about the failure location.

The TOP (Test Ordering Program) generates the path test sequence as the fail-and-branch tree structure and prepares necessary information to identify the failures.

The dictionary editor program edits the test dictionary based on the PE card placement dictionary and the output of the TOP to give details, such as the physical card position, card type, wiring between cards, etc., associated with the terminating point.

The contents of the dictionary are also merged to the tree-structure path test, and a path is picked from the failure detection path test string and translated into the PEXTAP input format by the failure location path test compilation program. Information to locate failures also appears in the document as a PEXTAP output.

The maintenance and modification of the path test will be continued to keep up with the design changes of the prototype and production PE's. A few additional programs will be required for running the path tests with the PE exerciser hooked to the supervising computer.

1.1.3.2 Combinational Tests

A reduced set of test patterns has been generated for the Carry Propagate Adder based on the new design.

1.1.3.3 Control Logic Tests

A set of Boolean equations of PE control signals was derived from the control logic wire list. This set of equations contains a significant amount of redundancy. During this period, a program was written to reduce redundant terms and apply DeMorgen's theorem to the equations. These simplified Boolean equations are transformed into a graph and then fed into PGM program. From the output of PGM, a minimal set

of tests to diagnose the control logic can be derived. The purpose for this transformation was also written in this period. The first section of the program is to find illegal expressions in the Boolean expressions; the second section is to rearrange signal names in order to fit PGM; the third is to expand the Boolean expressions inside the innermost set of parentheses into graph form and then to expand toward the outside set of parentheses. In the present version, about 100 Boolean equations are associated with the control logic, and they are transformed into a graph which contains about 1,800 card images. A minor modification will be made in order to increase the power of the program.

1.1.4 PEX Computer and Supervisor System

A modification of the PE Exerciser from the original manually-controlled papertape-based equipment to a high-speed flexible computer-controlled system had been proposed earlier by the University. The Burroughs programming group had joined in calling for the adding of additional control and computational capabilities to the PEX. However, in this quarter, the Burroughs Project management decided to terminate their efforts along this line. The University diagnostics group started designing both hardware and software for the control computer.

The PEX as currently implemented, although soundly designed, simply lacks the speed and flexibility necessary to insure rapid debugging of PE's. After ILLIAC IV was operational, this deficiency could have caused a higher than necessary MTTR, requiring the maintenance of additional spares and perhaps reducing the availability of ILLIAC IV. While ILLIAC IV is being produced, the deficiency would have been critical. It would have been difficult to meet current prototype debugging and production check-out schedules with an unaugmented PEX. Therefore, it was proposed that a small computer of the 16-18 bit word length class be interfaced to the PEX to be used as a dedicated PEX controller. Various candidate computer systems were evaluated and a Digital Equipment Corporation PDP-9/L computer was selected as offering the most flexible, lowest cost system which could be obtained within the necessary time frame.

Improvements to be wrought by this addition include the following:

(1) More rapid input to the PEX. As currently implemented, the PEX has only paper tape and cards as input media. The computer system includes magnetic tape; this will result in an order of magnitude increase in input rate.

(2) Greater control by technician. With the computer-controlled system, the technician will have control over the consequences of a test failure (e.g., print and continue, halt, branch, etc.). Further, the technician will be able to control repeated execution of the tests.

(3) Assembly and running of arbitrary groups of arbitrary tests. This will permit the technician to create his own group of tests, when and as needed, rather than be dependent only on pre-assembled routines.

(4) Greater information to technician. As currently implemented, the PEX only displays the differences between what was expected and what was obtained. With the computer system, the actual and expected results will both be printed, and the technician will be able to display or modify the contents of any register in the PE.

(5) Analysis of test results and dynamic modification and adaptation of tests. With the proposed system, the full 64-bit result will be sent to the computer for analysis. Interactive, on-line modification of tests in response to previous results can be implemented, permitting rapid isolation of the failure area and order-of-magnitude faster fault diagnosis.

(6) Assembly of PEXTAP programs. The assembler for compiling programs written in the PEX assembly language, PEXTAP, must be run on a B5500 or B6500. This function can ultimately be transferred to the PEX control computer, lessening the dependence of the diagnostic and maintenance group on outside facilities, particularly facilities being used to control ILLIAC IV.

A PEX Supervisor language had already been designed by Burroughs, with some modifications and improvements suggested by the Diagnostics Group at the University. Major elements of the supervisor have been analyzed and designed, and coding should be easily accomplished once the machine has arrived. While the language was being designed, close liaison was maintained with the technicians who will actually do the debugging of the

PE's. The proposed language has their enthusiastic support as an instrument which, when implemented through the desired computer system, will significantly increase their productivity.

1.1.5 CU Diagnostics

1.1.5.1 Bad Board Simulator

The specification of Bad Board (Card) Simulator has been defined and possible failure modes have been specified.

(1) Failure Modes: Failure modes were defined for each package based on the logic diagrams of DIL packages supplied by Burroughs. For the purpose of reducing the number of failure modes several considerations were made.

(a) Indistinguishable failure: Within a package some failures are indistinguishable: for example, for an AND gate input and the output stuck at zero, failures are indistinguishable. Also, if a logical element has only one fan-out, the failures of its output and of the input to the next element are indistinguishable. A class of all indistinguishable failures is represented by an equivalent representative failure.

(b) Undetectable failure: The logic equations in the Bad Board Simulator express all possible failures in a package. But all logic elements are not always used. Therefore, if a failure of an unused element is applied to Bad Board Simulator, a test for this failure cannot be found. The other case is that of an input pin connected to a constant voltage. This is equivalent to a stuck failure.

(c) Primary and Secondary failures: Some distinguishable and different failures can be detected by a test which detects other failures. Therefore, those failures are called secondary failures and these primary failures. Though both failures are described in the logic equations, the generation of test patterns will work only on primary failures at first. Thus, the number of failure modes to be considered is reduced. The average number of failure modes per package is about 15 to 20, and a typical CU Board contains about 1500 failure modes.

(2) Bad Board Simulator Generator: All parts of the CU card Simulator Generator System except the simulator body generation program can be applied to the Bad Board Simulator Generator. This program has been coded and is now being debugged. Some features of Bad Board Simulator are: (a) The Bad Board Simulator will accept any possible single logic failure. It will also simulate correct logic. (b) Since the program has the capability of parallel simulation, 47 different input patterns are simulated in parallel.

1.1.5.2 CU Card Test Generator

The final organization of the CU card test generator system has been investigated, during this period. The system will be composed of a Failure Mode Collector, Pattern Generator, and Test Controller into which the Good/Bad Card Simulator is to be incorporated, as well as other functions formerly implemented by other blocks. The coding and debugging of the Failure Mode Collector and the Pattern Generator have been completed.

At the present time, some parts of the Test Controller have been coded and debugged and it is expected that a first version of the system will soon be operational.

1.1.5.3 On-line Diagnostics

Preliminary drafts of Diagnostics Specifications have been received from Burroughs. The review of techniques and assignment of task items as well as rescheduling of implementation of the programs will occur in the middle of July.

1.2 Design Automation

Work has continued, at the University, on the Post-Processor Program. Small additions have been made to the software while the program is being tested in a production environment. Burroughs has been using the program, together with electrical draftsmen, to check PC artwork before release.

The plan to do Burroughs' DA computer runs on the University's B5500 is paying off. The early estimates for the computer time required have proven to be quite low. All PC artwork for the CU is being routed on our B5500.

2. SOFTWARE

2.1 Operating System

During this quarter, the design for the operating system was reviewed and modified. The system design philosophy remains oriented toward a highly distributed system that will be operationally efficient in a multiprogramming environment. The key to distributing control among the system elements is the use of buffer files. With these files one program can send a message to another, or several programs can contribute to the message. The buffer file mechanism was implemented on the B5500 during this quarter.

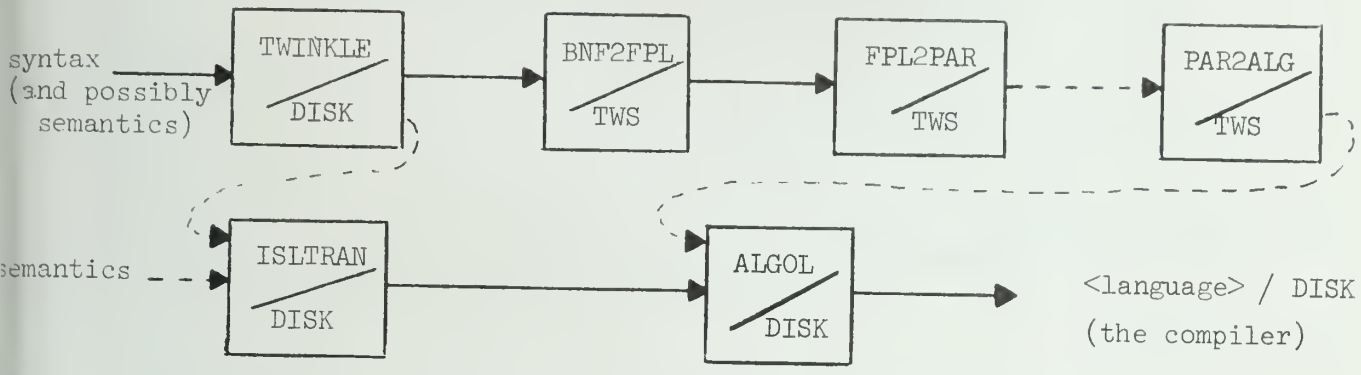
Another major effort of the quarter was the commencement of meetings between the operating system programmers and those working on Tranquil. The meetings have proved to be exceptionally fruitful and are occurring at a time when both segments of the projects are still malleable.

With minor deviations, the schedule given in the preceding QPR has been maintained. This schedule is still being followed.

2.2 Translator Writing System

A new method of making a Floyd production parser was conceived. This method uses a "Floyd production" procedure which is called once for each Floyd production with different parameters for each call. These parameters are switch indices and operands corresponding roughly to the individual parser pseudo-orders and their operands. The code for each pseudo-instruction is incorporated as a subcase in the Floyd production procedure, each executed according to the values of the switch indices. This method will result in one access of the parsing table per Floyd production instead of one access for each operator or operand. The result should be a faster parser since there will be less overlay and less calculation of parsing table indices. The direct (ALGOL) implementation will be blocked according to the logic structure of the Floyd production language and hence to the advantage of the B5500 rather than the artificial blocking now used. It will also be much smaller than the previous version. The result of both of the above differences is expected to be a direct implementation which is both fast enough and small enough to be practical for even medium sized languages.

The new version will also implement the first version of TWINKLE as a syntax metalanguage with possible automatic linking to ISLTRAN and to the ALGOL compiler. The sequence of programs executed are indicated by the following directed graph where "——→" indicates a necessary automatic step and "- - -→" indicates an optional automatic step.



The compilers generated are suffixed "DISK" and are used as in the following example for the language "DUMMY":

```
? USER = ME

? COMPILE MY/OBJCODE WITH DUMMY LIBRARY
                                (or syntax)

? DATA CARD
[
    source code

? END
```

During this quarter, a complete revision of the overall structure of the TWS system was made. The reasons for such revision were threefold:

(1) The TWINKLE compiler is ready to be integrated into the system. This asked for considerable modifications in the ISL translator and syntax preprocessor.

(2) As a result of the tests described in the last report which indicated the inefficiency of the directly coded (or executable) PARSER, a completely new way of generating such parsers was devised, as outlined above. These modifications required a complete rewriting of the parser-instructions generator and the converter from parser-instructions to ALGOL code.

(3) Since it is expected that the system will be in its final form by the end of the summer, several features, long planned but never implemented, are now being added. Among these are: (a) the ability to submit the syntax and semantics of a language together in one file and have the system end up with a compiler for that language, after automatically executing six separate programs; and (b) the provision to accept n-pass compilers ($n \geq 1$) in a completely automatic way.

The final structure of the system is now settled, and work in the modifications described is well advanced.

2.3 Compilers and Translators

2.3.1 TRANQUIL

Work on the pass 2 semantics ALGOL coding and debugging continued during this quarter.

A basic subset of the defined TRANQUIL language (which will hopefully be implemented by October 1, 1969) has been identified, and plans and schedules were made for completing this subset. Some of the features not included in this basic subset are: array operators, multidimensional sets, procedures, I/O, simultaneous control using general sets (GENSETS) and 32-bit arithmetic.

Procedures and a macro facility for TRANQUIL have been specified. Implementation of procedures for TRANQUIL has been initiated at the pass 1 level.

The TRANQUIL compiler group has attempted to improve communications both with prospective users of TRANQUIL and with the operating system designers and implementors. Discussions with users have led to specifications for compiler diagnostics and debugging aids. The TRANQUIL-operating system interface requirements have been investigated, particularly the areas of storage allocation, I/O, and dynamic overlay of data and code. Work in these areas will continue.

Considerable time has been devoted this quarter to the problems of work load analysis, scheduling, and staffing for implementation and maintenance of the compiler beyond the basic subset.

2.3.2 GLYPNIR

GLYPNIR is now fully implemented as described in DCS Report 332, except for certain literals and the ABS, SIGN, and ENTIER functions. Further development of the language will be delayed pending user evaluation of the current package.

2.4 Assembler

No work was done on the ILLIAC IV assembler during the last quarter due to commitments of personnel to the DA and Diagnostics areas of the project. It is expected that this investment will pay off by reducing the execution time of some of the DA/Diagnostics programs and thereby free time for debugging the operating system.

2.5 Simulator

During this quarter, several errors were found and corrected in the ILLIAC IV execution simulator. A facility for ascertaining PE usage, i.e., the number of PE's enabled, and for summarizing it was added to the execution and timing simulators. The clock time in the timing simulator was changed from 40 nsec to 50 nsec.

2.6 B5500 Operation

The present status of the MCP is Mark IX, LEVEL 31.14. During April 184.97 hours of processor time were logged. The comparable figure for May is 148.97 hours. Tape storage cabinets and a card file are being ordered for the Tape Documentation library. There are operators on duty during all scheduled operating hours.

3. APPLICATIONS

3.1 Numerical Analysis

3.1.1 Monte Carlo Transport

The Monte Carlo Transport code was expanded to include one space dimension and three angular dimensions. Problems involving absorption and scattering were compared with closed form solutions given in Chandasekar, and the results were quite good for the number of particles used. There are several ways this program can be expanded to two spatial dimensions and formulated for ILLIAC IV. Among the most prominent alternatives are:

1) Assign a particle uniquely to a PE. This requires that either all cross sections must be stored in a PE or the cross sections must "migrate" among the PE's.

2) Assign a PE to a unique section of the space mesh. This implies that the particles must "migrate". Efficiency requires that the number of particles stored in PE_i be approximately the same as the number stored in PE_j.

3) Assign a PE to a unique section of the space mesh and require that the number of particles handled by each PE be the same. Differences in flux and intensity must then be reflected by particle "weighting".

Analysis of these and other schemes will continue. It is unlikely that the same scheme will be optimum for both linear and nonlinear transport, particularly if fluid motion of the medium is also included.

3.1.2 Multidimensional Compressible Hydrodynamics

Hick's test problems are still being run and plotted on the 360/75 to test various velocity weighting schemes. An interesting pattern is occurring. At the shock front, the "donor" cell scheme gives excellent results--far superior to the other two schemes. But away from the shock front, where the flow is smooth, the "donor" scheme breaks down, and the "OIL" velocity weighting is superior. No scheme works very satisfactorily during the collision of two shocks.

A two dimensional code has been written for the B5500, and properties of the three velocity weighting schemes will be examined in two dimensions.

3.1.3 Polynomial Root Finding

During this quarter, the algorithm for evaluating

$$\int_S \frac{P_n'(z)}{P_n(z)}\,dz \quad ,$$

where the contour S is a square in the complex plane, was tested on the B5500. It was desired to find out how many subdivisions were required in the numberical quadrature, if more accurate methods (Simpson rule or Gaussian Quadrature) were faster, and how near to the contour a root could lie without causing the integration to diverge.

Other strategies for refining roots were also considered. If a root is contained in a square whose dimension is s, then by subdividing it into 64 subsquares and allowing each PE to simultaneously integrate around a subsquare, the uncertainty in the root is reduced to 1/8 s. But using a single processor, the same reduction in the uncertainty can be achieved in no more than 12 quadratures. This suggests that the "efficiency" of the procedure is less than 100% even though all PE's are busy. A more effective strategy is to assign each PE to a root (or divide the PE's in some regular fashion if there are less than 64 roots isolated) and this is being investigated.

3.1.4 Seismic Equation of State

During this quarter, an attempt was made to program an "equation of state" for earth. The thermodynamic equation of state (EOS) gives pressure, P, as a function of density, ρ, and internal energy, E. The EOS is obtained by "fitting" experimental data with a certain function. Unfortunately, this function has a different form over different ranges of ρ, E.

Each PE is considered to contain a value for ρ, and a value for E (64 two-tuples). The subscript on the P's in the following equations represents the function which must be evaluated when the (ρ, E) obeys

the conditions

(I) for $\rho \geq \rho_0 = 2.6$

$$P_S = \left[a + \frac{b}{\frac{E}{E_0\eta^2} + 1}\right] \cdot E\rho + A\mu + B\mu^2 \tag{1}$$

(II) for $\rho < \rho_0 = 2.6$, $E < E_S$,

(a) if $\rho < 2.40214$, then

$$P_1 = -0.740 \left\{ \frac{(\mu + 0.385)[1 + 105(\mu + 0.0761)^2]}{(\mu + 0.0406)[1 + 623(\mu + 0.0761^2]} \right\} \times 10^9 + 0.8624035819 \times 10^6 \tag{2}$$

(b) if $2.40214 \leq \rho < \rho_0$, then

$$P_2 = (\mu + 0.1073)[170 + 1{,}140(\mu + 0.1073)] \times 10^9 + P_{ATM} \tag{3}$$

(III) for $\rho < \rho_0$, $E > E_S$

$$P_V = aE\rho + \left[\frac{bE\rho}{\frac{E}{E_0\eta^2} + 1} + A\rho\, e^{-\beta\left(\frac{V}{V_0} - 1\right)}\right] e^{-\alpha\left(\frac{V}{V_0} - 1\right)} \tag{4}$$

(IV) for $\rho < \rho_0$, $E_S < E < E'_S$

$$P_{S+V} = \frac{1}{E'_S - E_S}\left[(E - E_S)\, P_V + (E'_S - E)P_S\right]$$

in which P_S and P_V must be entered as positive, and they will be set to P_{ATM} if those are negative. (In the above equations, μ, ρ, η, E, V, are parameters, and the others are constants.) The domain (ρ, E) of the conditions is sketched in Figure 1.

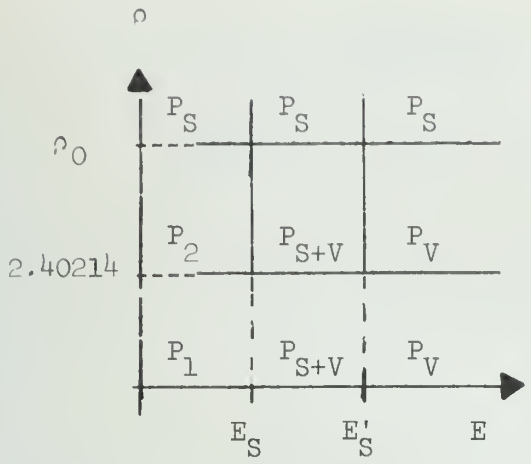


Figure 1

In order to achieve effective parallel computation, the concepts of "factoring" and "embedding" are used. The performance of similar arithmetic operations in different equations at the same time is sought. The similar sequence of arithmetic operations if "factored" or "extracted" as a symbolic form, for example, CONSTANT1 × (PARAMETER + CONSTANT2)^2 commonly occurs in all equations. If

(PARAMETER, CONSTANT1, CONSTANT2) → $(\mu, B, 0)$ for (1)

→ $(\mu, 77.7 \times 10^9, 0.0761)$ for (2)

→ $(\mu, 1{,}140 \times 10^9, 0.1073)$ for (3)

then is assigned to the three-tuple,

$$B\mu^2,\ 77.7 \times 10^9 \times (\mu + 0.0761)^2,\ 1{,}140 \times 10^9\ (\mu + 0.1073)^2$$

can be obtained, at the same time, as a portion of P_S, P_1 and P_2, respectively. For the above parallel computation, the memory allocation for CONSTANT1 and CONSTANT2 for those PE's which need to compute P_S, P_1, P_2 is seen in Figure 2.

	P_S	P_1	P_2	
CONST1	B	77.7×10^9	$1{,}140 \times 10^9$	← PE Memory
CONST2	0	0.0761	0.7073	

Figure 2

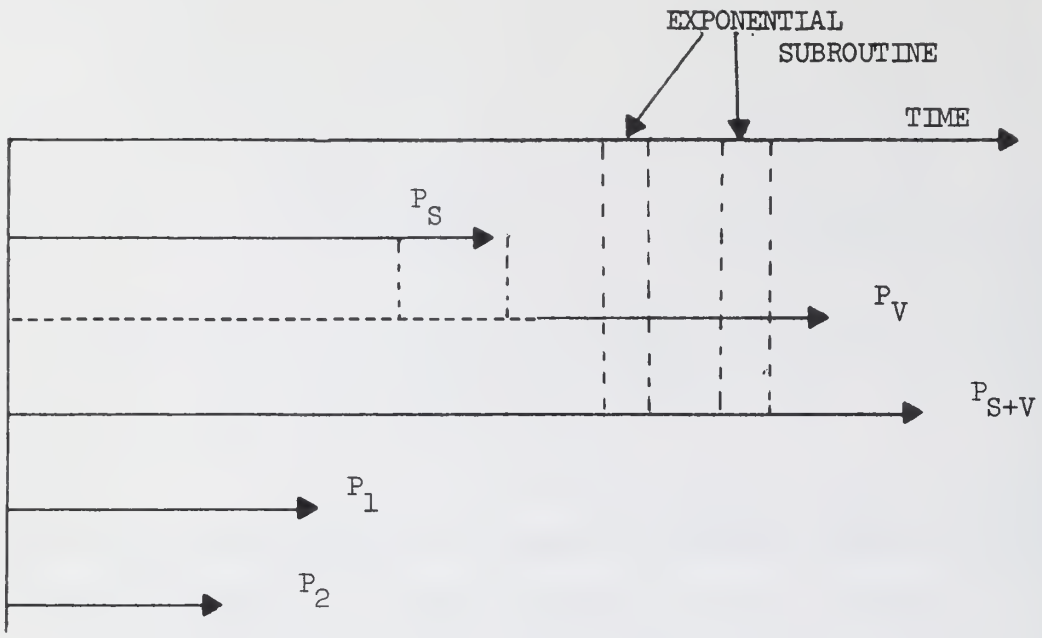


Figure 3

All of P_1 and P_2 can be calculated by complete "embedding" in P_S. Some parts of P_V, which are common P_S, start around the end of P_S. P_S and P_{S+V} are required to call exponential subroutine twice as shown in the figure. A more detailed description will be given in a forthcoming document.

This EOS has been programmed and simulated. It is estimated that no more than 80μs are required to evaluate the EOS for 64 separate two-tuples (ρ,E). If no member of the 64 two-tuples requires evaluation of P_V or P_{S+V}, the time is considerably shorter.

3.1.5 Mathematical Subroutines

The multiple precision divide routine was implemented during this quarter. Now all the arithmetic operations are available as multiple precision subroutines. Work is continuing on the transcendental and trigonometric function subroutines.

3.2 Linear Programming

During this quarter, SSK simulation of the LPS (small linear programming system) code was initiated. Because of the complexity of

the intermediate files, the group intends to conduct a sequential simulation of the routines involved. Thus, each procedure will utilize output from preceding routines as input instead of independently generated data. This will serve as a check on the validity of files produced in each routine as well as eliminate the extra coding and debugging requirements of a necessarily complex intermediate file generator. However, where input requirements are small, routines will be pretested with independent data in order to expedite the main "front end" simulation.

The first segment of LPS is the B6500 preprocessor. Input data must be converted from the user's format and "strewed" across the PEM's in ILLIAC IV disk records. Several interralated disk files (A-matrix, cost row, basis header, right hand side) are initialized here for final preprocessing in PEM's by ILLIAC IV itself. In addition, B6500 tables must be prepared for use in the postprocessing of the solution by the B6500. A preliminary version of a B5500 preprocessor has been completed and largely debugged. The assembly language routine for final preprocessing in PEM has been coded and simulation has begun.

The analysis of stability conditions and techniques for improving accuracy continues. It has been determined experimentally that forcing all bounds to the common value of 1 may have an adverse effect on the sizes of the resulting matrix coefficients. For this reason, different formulas for computing a dynamic, data-based value for the common upper bound are being investigated. For control and prediction of round-off errors, an iterative refinement technique and an L4 reinversion algorithm have been defined. The evaluation of these proposals will continue into the next quarter.

3.3 Long Codes

This quarter was devoted to the study of stochastic processes and the identification problem of linear stationary dynamic systems. The work done by Ho and Lee [1] and Lee [2] in this area was extended to treat a more general case than the one they presented.

Ho and Lee considered the system

$$x_{k+1} = \Phi x_k \qquad (1)$$

$$Z_k = h^t x_k + v_k$$

where x is the state vector and is of order n; Φ is the n x n transition matrix of the system; Z is the observed scalar; h^t is a row vector of order n; and v_k is a random noise sequence that cannot be measured and is assumed to be Gaussian.

Assuming system (1) to be completely observable [2, 3] and n-identifiable [2], where n is the order of the system, then it was possible to reduce the system to the canonical form [3],

$$y_{k+1} = \left[\begin{array}{c|c} o & I \\ \hline \multicolumn{2}{c}{\varphi^t} \end{array}\right] y_k$$

$$Z_k = (1, 0, \ldots, 0)\, y_k + v_k$$

by the linear transformation $y_k = O\, x_k$, where φ^t is a row vector of order n, and O is the observability matrix of the system.

A least-squares estimate of the vector φ was obtained and was updated as new measurements were observed. Convergence of this stochastic estimation was proved under certain plausible assumptions [1, 2].

Lee's work was extended in an effort to identify the system

$$\begin{aligned} x_{k+1} &= \Phi\, x_k \\ Z_k &= H\, x_k + v_k \end{aligned} \qquad (2)$$

where H is an m x n matrix.

As before, assuming that the system is completely observable and n-identifiable, it was possible to reduce system (2) to a canonical form by means of the linear transformation

$$y_k = B^{-1} x_k$$

where $B = \left[\begin{array}{c|c|c|c} x_o & \Phi\, x_o & \ldots. & \Phi^{n-1} x_o \end{array}\right]$ is the n-identifiability matrix, which is of rank by virtue of the above assumptions. A least-squares estimate of the vector φ representing the reduced n-unknowns is obtained. In obtaining that estimate, however, care must be taken that the observations

are spaced in batches of certain size to avoid correlated noise. The identification algorithm of system (2) will be discussed in detail in a forthcoming ILLIAC IV document.

3.4 Radar Data Processing

The Kalman Filtering Tracking program has been completed. It was written in ILLIAC IV Assembly Language using an algorithm developed at Lincoln Laboratory. All operations are done in the 32-bit mode, and all data is represented in Radar Polar coordinates. The whole program has been run on the simulator, and simulated ILLIAC IV computer results have been produced. A document explaining this program and its results is now being written. It should be completed by the end of the next quarter.

The FFT (Fast Fourier Transform) algorithm has been programmed in ILLIAC IV Assembly Language using 32-bit mode. The real and the imaginary parts of a FFT occupy one 64-bit word in each PE; therefore, both are treated simultaneously. A report on this algorithm is presently being generated.

3.5 Seismic Signal Processing

A significant portion of the quarter was spent modifying the statistical programs written during the previous quarter, doing the research necessary for designing digital filters for seismic signal processing, and building a signal processing library system on tape.

The frequency analysis system begun during the last quarter was completed. The system now has the capability of deriving the Fourier Amplitude spectrum of a given time series through the use of the sine and cosine transform or through the Fast Fourier Transform program made available from the Phased Array Radar group.

Some program modifications have been made to improve the data transmission to the various programs and create disk files of data for manipulation by the various seismic system programs. The convolution program has been redesigned to employ the concept of polynomial multiplication used as a transform. This will be useful in future applications. The programs are written in B5500 ALGOL for later conversion to the desired ILLIAC IV language.

3.6 Statistical Packages

During this quarter, the design for the ILLIAC IV statistical system was altered considerably due to personnel restrictions. Before this time, a system with considerable innovation was under design. It has been decided now, however, that the system should be limited to the very basic tools and that no experimentation with input language should be attempted at this time. Therefore, the design will mimic the design of the University of Illinois SOUPAC system originally written on the 7094 and converted to and updated for the IBM 360/75.

The routines which will decode the input language will run on the B6500. The coding of these routines was virtually completed during this quarter. They have also been partially debugged on the B5500. It has been finally decided to write the routines themselves in TRANQUIL. The routines will attempt, however, to use only the simplest features of the language consistent with efficiency on ILLIAC IV. Rough coding and flowcharting has begun on the correlation program. The structure of all the routines themselves will follow the structure of the same programs on the SOUPAC system with the exception of the transformations program and the matrix program. The matrix program on ILLIAC IV will simply access the standard matrix written on ILLIAC IV. The transformations program and the matrix program are under consideration. It will mimic the operations of the SOUPAC system but will be simplified.

3.7 Burroughs ALGOL to Control Data Corp. 6000 Series ALGOL Conversion

The ALGOL to ALGOL conversion project progressed in two major ways this quarter.

(1) Neglecting input-out related phenomena, stream procedures, and certain items noted in the following section, the ALGOL to ALGOL conversion program can translate working Burroughs ALGOL programs into working CDC ALGOL programs. This was demonstrated by having itself to produce what, with the exceptions previously mentioned, was virtually confirmed to be a syntactically correct, equivalent CDC ALGOL program. This conversion program runs a 600 cards per processor minute.

(2) Computer time on a CDC 6000 series machine was finally obtained at Northwestern University near Chicago. Runs were made on this CDC 6400 to determine what machine code was generated by the CDC ALGOL Compiler for various ALGOL constructs.

CDC ALGOL is almost true ALGOL 60. It uses a general stacking algorithm that is not troubled by the up level addressing problem as is Burroughs ALGOL on the B5500. Also, this ALGOL makes considerable use of descriptors as parameters so that run time attribute testing and conversion (between real and integer only) may be done. Unfortunately, this quite general data structure requires execution time. Tests have demonstrated that simple ALGOL constructs execute in the same, if not more, processor time on the 6400 as on the B5500, even though the 6400 is a considerably faster machine with many more active registers. Procedure cells require five times more processor time on the 6400 than on the B5500.

It was discovered that CDC ALGOL evaluated expressions with no consideration of side effects. These occur frequently in Burroughs ALGOL programs due to the heavy use of imbedded assignments (which are converted to function calls by the conversion program). Runs were made on the 6400 to determine precisely what code was generated for various types of expressions. The conversion program was changed to modify expressions so that the code generated by CDC ALGOL evaluates the elements of an expression such that side effects propagate in the manner intended by the Burroughs ALGOL programmer.

Burroughs ALGOL permits using a sub-array of an array as a procedure argument. The conversion program and the CDC ALGOL utility routines are being modified to simulate this construct.

Various machine language routines for partial word operators, concatenate operators, etc., must be written and debugged. An I/O package must be designed and implemented.

3.8 ILLIAC IV Education

The ILLIAC IV Education courses, CS 491-D and B5500 ALGOL, were successfully completed this quarter. As these courses are again offered, modifications and improvements will be made to them.

The paper describing the contents of the Graduate level CS 491-D, which covered ILLIAC IV hardware, software, and applications was also completed

this quarter. It was written from notes taken in these classes, and in some cases, the lectures were summarized and dictated for the purpose of having them included in the paper.

The plans for creating a program library were modified to include documents, reports, and papers written about ILLIAC IV. This will now be developed as a users library and will be implemented in the coming quarter.

[1] Ho, Y. C., and Lee, R. C. "Identification of Linear Dynamic Systems," Information and Control, VIII (1965), 93-110.

[2] Lee, R. C. "Optimal Estimation, Identification and Control," Research Monograph, XXVIII, Cambridge, Massachusetts: M.I.T. Press, 1964.

[3] Bucy, R. S., and Joseph, R. D. "Filtering for Stochastic Processes with Applications to Guidance," Interscience Tracts in Pure and Applied Mathematics, XXIII, Interscience Publishers: J. Wiley, 1968.

THESES

Bernott, Bruce A. "Disk I/O for Non-Core-Contained P.D.E. Meshes and Arrays." Master's thesis, DCS Report No. 311. Urbana, Illinois: Department of Computer Science, University of Illinois, 1969.

Lawrie, Duncan H. "GLYPNIR: A List Processing Language for ILLIAC IV." Master's thesis, DCS Report No. 322. Urbana, Illinois: Department of Computer Science, University of Illinois, 1969.

Matsushita, Yutaka. "A Solution to the Hidden Line Problem." Master's thesis, DCS Report No. 335. Urbana, Illinois: Department of Computer Science, University of Illinois, 1969.

McCarthy, Thomas E. "An Automatic Integrator for Ordinary Differential Equations for ILLIAC IV." Master's thesis, DCS Report No. 336. Urbana, Illinois: Department of Computer Science, University of Illinois, 1969.

DOCUMENTS

Grossman, Gary R. "SSK/SSK Reference Manual." ILLIAC IV Doc. No. 217, DCS File No. 798, (June 13, 1969).

Northcote, Robert S., et al. "TRANQUIL: A Language for an Array Processing Computer." ILLIAC IV Doc. No. 215, (April 1, 1969).

Winje, Gilbert L. "Random Number Generators for ILLIAC IV." ILLIAC IV Doc. No. 218, DCS File No. 799, (June 20, 1969).

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*

Department of Computer Science
University of Illinois
Urbana, Illinois 61801

2a. REPORT SECURITY CLASSIFICATION

UNCLASSIFIED

2b. GROUP

3. REPORT TITLE

ILLIAC IV QUARTERLY PROGRESS REPORT
April, May, June 1969

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

April - June, 1969 - Progress Report of the ILLIAC IV Project

5. AUTHOR(S) *(First name, middle initial, last name)*

6. REPORT DATE

August 1, 1969

7a. TOTAL NO. OF PAGES

31

7b. NO. OF REFS

10

8a. CONTRACT OR GRANT NO.

46-26-15-305

b. PROJECT NO.

USAF 30(602)-4144

c.

d.

9a. ORIGINATOR'S REPORT NUMBER(S)

ILLIAC IV Document No. 230
DCS Report No. 360

9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*

RADC TR

10. DISTRIBUTION STATEMENT

Qualified requesters may obtain copies of this report from DCS.

11. SUPPLEMENTARY NOTES

NONE

12. SPONSORING MILITARY ACTIVITY

Rome Air Development Center
Griffis Air Force Base
Rome, New York 13440

13. ABSTRACT

See the Report Summary on Page 1 within the Report itself.

DD FORM 1 NOV 65 1473

UNCLASSIFIED

Security Classification

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
PE Logic Debugging and Diagnostics						
CU Logic Simulator System						
TESLA						
Assembled Code Translator (ACT III)						
PEX						
CU Diagnostics						
CU Card Test Generator						
On-line Diagnostics						
Design Automation						
Operating System						
Translator Writing System						
Compilers and Translators						
TRANQUIL						
GLYPNIR						
Assembler						
Simulator						
B5500						
Numerical Analysis						
Monte Carlo Transport						
Multidimensional Compressible Hydrodynamics						
Polynomial Root Finding						
Seismic Equation of State						
Linear Programming						
Long Codes						
Radar Data Processing						
Seismic Signal Processing						
Statistical Packages						
Burroughs ALGOL to Control Data Corporation 6000 Series ALGOL Conversion						
ILLIAC IV Education						